Terms used **multi processor exception handling** Found **52,318** of **150,138**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

**1** The use of multithreading for exception handling
Craig B. Zilles, Joel S. Emer, Gurindar S. Sohi
November 1999 **Proceedings of the 32nd annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.49 MB) Publisher Site Additional Information: full citation, abstract, references, citings, index terms

Common hardware exceptions, when implemented by trapping, unnecessarily serialize program execution in dynamically scheduled superscalar processors. To avoid the consequences of trapping the main program thread, multithreaded CPUs can exploit control and data independence by executing the exception handler in a separate hardware context. The main thread doesn't squash instructions after the excepting instruction, conserving fetch bandwidth and allowing execution of instructions inde ...

**2** Hardware and software support for efficient exception handling
Chandramohan A. Thekkath, Henry M. Levy
November 1994 **Proceedings of the sixth international conference on Architectural support for programming languages and operating systems**, Volume 29 , 28 Issue 11 , 5
Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Program-synchronous exceptions, for example, breakpoints, watchpoints, illegal opcodes, and memory access violations, provide information about exceptional conditions, interrupting the program and vectoring to an operating system handler. Over the last decade, however, programs and run-time systems have increasingly employed these mechanisms as a performance optimization to detect normal and expected conditions. Unfortunately, current archi ...

**3** An architectural framework for migration from CISC to higher performance platforms
Gabriel M. Silberman, Kemal Ebcioğlu
August 1992 **Proceedings of the 6th international conference on Supercomputing**
Full text available: pdf(2.04 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a novel architectural framework that allows software applications written for a given Complex Instruction Set Computer (CISC) to migrate to a different, higher performance architecture, without a significant investment on the part of the application user or developer. The framework provides a hardware mechanism for seamless switching between two instruction sets, resulting in a machine that enhances application performance while keeping the same program behavior (from a user per ...

4 Balancing register allocation across threads for a multithreaded network processor
Xiaotong Zhuang, Santosh Pande
June 2004 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation**, Volume 39 Issue 6
Full text available: pdf(429.85 KB) Additional Information: full citation, abstract, references, index terms

Modern network processors employ multi-threading to allow concurrency amongst multiple packet processing tasks. We studied the properties of applications running on the network processors and observed that their imbalanced register requirements across different threads at different program points could lead to poor performance. Many times application needs demand some threads to be more performance critical than others and thus by controlling the register allocation across threads one could impa ...

**Keywords**: multithreaded processor, network processor, register allocation

5 On high-bandwidth data cache design for multi-issue processors
Jude A. Rivers, Gary S. Tyson, Edward S. Davidson, Todd M. Austin
December 1997 **Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture**
Full text available: pdf(1.38 MB) Additional Information: full citation, abstract, references, citings, index terms
Publisher Site

Highly aggressive multi-issue processor designs of the past few years and projections for the decade, require that we redesign the operation of the cache memory system. The number of instructions that must be processed (including incorrectly predicted ones) will approach 16 or more per cycle. Since memory operations account for about a third of all instructions executed, these systems will have to support multiple data references per cycle. In this paper, we explore reference stream characterist ...

**Keywords**: Locality-Based Interleaving, Multiporting, High-Bandwidth Data Supply, Multi-Bank Caches

6 On micro-kernel construction
J. Liedtke
December 1995 **ACM SIGOPS Operating Systems Review , Proceedings of the fifteenth ACM symposium on Operating systems principles**, Volume 29 Issue 5
Full text available: pdf(1.65 MB) Additional Information: full citation, references, citings, index terms

7 Dynamic speculation and synchronization of data dependences
Andreas Moshovos, Scott E. Breach, T. N. Vijaykumar, Gurindar S. Sohi
May 1997 **ACM SIGARCH Computer Architecture News , Proceedings of the 24th annual international symposium on Computer architecture**, Volume 25 Issue 2
Full text available: pdf(2.51 MB) Additional Information: full citation, abstract, references, citings, index terms

Data dependence speculation is used in instruction-level parallel (ILP) processors to allow early execution of an instruction before a logically preceding instruction on which it may be data dependent. If the instruction is independent, data dependence speculation succeeds; if not, it fails, and the two instructions must be synchronized. The modern dynamically scheduled processors that use data dependence speculation do so blindly (i.e., every load instruction with unresolved dependences is spec ...

8 The design of a virtual machine for Ada
L. J. Groves, W. J. Rogers

November 1980 **ACM SIGPLAN Notices , Proceeding of the ACM-SIGPLAN symposium on Ada programming language**, Volume 15 Issue 11

Full text available: pdf(1.27 MB) Additional Information: full citation, abstract, references

An implementation of Ada should be based on a machine-independent translator generating code for a Virtual Machine, which can be realised on a variety of machines. This approach, which leads to a high degree of compiler portability, has been very successful in a number of recent language implementation projects and is the approach which has been specified by the U. S. Army and Air Force in their requirements for Ada implementations.This paper discusses the rationale, requirements and design of s ...

9 The operating system and language support features of the BELLMACTM-32 microprocessor.

Alan D. Berenbaum, Michael W. Condry, Priscilla M. Lu

March 1982 **Proceedings of the first international symposium on Architectural support for programming languages and operating systems**, Volume 10 , 17 Issue 2 , 4

Full text available: pdf(738.86 KB) Additional Information: full citation, abstract, references, citings, index terms

The BELLMAC-32 microprocessor is a 32-bit microprocessor, implemented with CMOS technology, designed to support operating system functions and high level languages efficiently. The architecture was designed with the following objectives in mind: • High performance. • Enhanced operating system support capabilities. • High level language support. • High reliability, availability and maintainability.

10 Overcoming the limitations of conventional vector processors

Christos Kozyrakis, David Patterson

May 2003 **ACM SIGARCH Computer Architecture News , Proceedings of the 30th annual international symposium on Computer architecture**, Volume 31 Issue 2

Full text available: pdf(160.23 KB) Additional Information: full citation, abstract, references

Despite their superior performance for multimedia applications, vector processors have three limitations that hinder their widespread acceptance. First, the complexity and size of the centralized vector register file limits the number of functional units. Second, precise exceptions for vector instructions are difficult to implement. Third, vector processors require an expensive on-chip memory system that supports high bandwidth at low access latency.This paper introduces CODE, a scalable vector ...

11 Multi-dimensional resource scheduling for parallel queries

Minos N. Garofalakis, Yannis E. Ioannidis

June 1996 **ACM SIGMOD Record , Proceedings of the 1996 ACM SIGMOD international conference on Management of data**, Volume 25 Issue 2

Full text available: pdf(1.47 MB) Additional Information: full citation, abstract, references, citings, index terms

Scheduling query execution plans is an important component of query optimization in parallel database systems. The problem is particularly complex in a shared-nothing execution environment, where each system node represents a collection of time-shareable resources (e.g., CPU(s), disk(s), etc.) and communicates with other nodes only by message-passing. Significant research effort has concentrated on only a subset of the various forms of intra-query parallelism so that scheduling and synchronizati ...

12 Reliability Issues in Computing System Design

B. Randell, P. Lee, P. C. Treleaven

June 1978 **ACM Computing Surveys (CSUR)**, Volume 10 Issue 2

Full text available: pdf(3.95 MB) Additional Information: full citation, references, citings, index terms

13

Exception handling in communication protocols
M. Stella Atkins
October 1983 **ACM SIGCOMM Computer Communication Review , Proceedings of the eighth symposium on Data communications**, Volume 13 Issue 4
Full text available: pdf(488.48 KB) Additional Information: full citation, abstract, references, index terms

This paper describes structures used for exception handling in multi-process implementations of the X.25 and X.29 protocols. Each protocol layer is implemented as a collection of cooperating processes. These are structured in one of two alternative ways, reflecting differences in the type of exceptional situations that may occur. Within each of the lower protocol layers, one process takes the form of a Finite State Machine. In these lower level protocols, where it is ...

14 Fast barrier synchronization hardware
Carl J. Beckmann, Constantine D. Polychronopoulos
November 1990 **Proceedings of the 1990 ACM/IEEE conference on Supercomputing**
Full text available: pdf(984.65 KB) Additional Information: full citation, abstract, references

Many recent studies have considered the importance of barrier synchronization overhead on parallel loop performance, especially for large-scale parallel machines. This paper describes a hardware scheme for supporting fast barrier synchronization. It allows barrier synchronization to be performed within a single instruction cycle for moderately sized systems, and is scalable with logarithmic increase in synchronization time. It supports a large number of concurrent barriers, and can also be used ...

15 Threads and input/output in the synthesis kernal
H. Massalin, C. Pu
November 1989 **ACM SIGOPS Operating Systems Review , Proceedings of the twelfth ACM symposium on Operating systems principles**, Volume 23 Issue 5
Full text available: pdf(1.34 MB) Additional Information: full citation, abstract, references, citings, index terms

The Synthesis operating system kernel combines several techniques to provide high performance, including kernel code synthesis, fine-grain scheduling, and optimistic synchronization. Kernel code synthesis reduces the execution path for frequently used kernel calls. Optimistic synchronization increases concurrency within the kernel. Their combination results in significant performance improvement over traditional operating system implementations. Using hardware and software emulating a SUN 3 ...

16 Overview of concert multilisp: a multiprocessor symbolic computing system
Robert H. Halstead
March 1987 **ACM SIGARCH Computer Architecture News**, Volume 15 Issue 1
Full text available: pdf(638.45 KB) Additional Information: full citation, abstract, citings, index terms

Multilisp is a parallel programming language derived from the Scheme dialect of Lisp by addition of the future construct. Multilisp has been implemented on Concert, a shared-memory muitiprocessor that uses a novel *RingBus* interconnection. Concert currently has 28 MC68000 processors, with a design goal of 32 processors. Several application programs have been developed and measured using Concert Multilisp. Experience with these programs has contributed to tuning the Multilisp language desig ...

17 Fault-tolerance in the advanced automation system
Flaviu Cristian, Bob Dancey, Jon Dehn
September 1990 **Proceedings of the 4th workshop on ACM SIGOPS European workshop**
Full text available: pdf(1.39 MB) Additional Information: full citation, abstract, references, citings

The Advanced Automation System is a distributed real-time system under development by IBM's Systems Integration Division for the US Federal Aviation Administration. The system is intended to replace the present en-route and terminal approach US air traffic control

computer systems over the next decade. High availability of air traffic control services is an essential requirement of the system. This paper discusses the general approach to fault-tolerance adopted in AAS, by reviewing some of the q ...

18 A general framework for prefetch scheduling in linked data structures and its application to multi-chain prefetching

Seungryul Choi, Nicholas Kohout, Sumit Pamnani, Dongkeun Kim, Donald Yeung
May 2004 **ACM Transactions on Computer Systems (TOCS)**, Volume 22 Issue 2

Full text available: pdf(2.45 MB) Additional Information: full citation, abstract, references, index terms

Pointer-chasing applications tend to traverse composite data structures consisting of multiple independent pointer chains. While the traversal of any single pointer chain leads to the serialization of memory operations, the traversal of independent pointer chains provides a source of memory parallelism. This article investigates exploiting such *interchain memory parallelism* for the purpose of memory latency tolerance, using a technique called *multi--chain prefetching*. Previous work ...

**Keywords**: Data prefetching, memory parallelism, pointer-chasing code

19 Medusa: an experiment in distributed operating system structure

John K. Ousterhout, Donald A. Scelza, Pradeep S. Sindhu
February 1980 **Communications of the ACM**, Volume 23 Issue 2

Full text available: pdf(1.25 MB) Additional Information: full citation, references, citings

**Keywords**: deadlock, distributed systems, exception reporting, message systems, operating systems, task forces

20 VM-based shared memory on low-latency, remote-memory-access networks

Leonidas Kontothanassis, Galen Hunt, Robert Stets, Nikolaos Hardavellas, Michał Cierniak, Srinivasan Parthasarathy, Wagner Meira, Sandhya Dwarkadas, Michael Scott
May 1997 **ACM SIGARCH Computer Architecture News , Proceedings of the 24th annual international symposium on Computer architecture**, Volume 25 Issue 2

Full text available: pdf(1.96 MB) Additional Information: full citation, abstract, references, citings, index terms

Recent technological advances have produced network interfaces that provide users with very low-latency access to the memory of remote machines. We examine the impact of such networks on the implementation and performance of software DSM. Specifically, we compare two DSM systems---Cashmere and TreadMarks---on a 32-processor DEC Alpha cluster connected by a Memory Channel network.Both Cashmere and TreadMarks use virtual memory to maintain coherence on pages, and both use lazy, multi-writer releas ...

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Your search matched **6** of **1123491** documents.
A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

**Refine This Search:**
You may refine your search by editing the current search expression or entering a new one in the text box.

multi<and>processor<and>exception <and>handling Search

■ Check to search within this result set

**Results Key:**
**JNL** = Journal or Magazine **CNF** = Conference **STD** = Standard

---

1 **Concurrent event handling through multithreading**
*Kekckler, S.W.; Chang, A.; Chatterjee, W.S.L.S.; Dally, W.J.;*
Computers, IEEE Transactions on , Volume: 48 , Issue: 9 , Sept. 1999
Pages:903 - 916

[Abstract] [PDF Full-Text (288 KB)] **IEEE JNL**

---

2 **Multi Windows: a dynamic register array concept for high-performance RISC processors**
*Scholz, T.; Schafers, M.;*
EUROMICRO 94. System Architecture and Integration. Proceedings of the 20th EUROMICRO Conference. , 5-8 Sept. 1994
Pages:523 - 530

[Abstract] [PDF Full-Text (600 KB)] **IEEE CNF**

---

3 **Hardware implementation of a nonlinear processor**
*Jain, V.K.; Shrivastava, S.; Snider, A.D.; Damerow, D.; Chester, D.;*
Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on , Volume: 6 , 30 May-2 June 1999
Pages:509 - 514 vol.6

[Abstract] [PDF Full-Text (384 KB)] **IEEE CNF**

---

4 **An improved dynamic register array concept for high-performance RISC processors**
*Scholz, T.; Schafers, M.;*
System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on , Volume: 1 , 3-6 Jan. 1995
Pages:181 - 190 vol.1

[Abstract] [PDF Full-Text (676 KB)] **IEEE CNF**

---

5 **The use of multithreading for exception handling**
*Zilles, C.B.; Emer, J.S.; Sohi, G.S.;*
Microarchitecture, 1999. MICRO-32. Proceedings. 32nd Annual International Symposium on , 16-18 Nov. 1999
Pages:219 - 229

[Abstract] [PDF Full-Text (72 KB)] **IEEE CNF**

---

6 **Error detection and handling in a superscalar, speculative out-of-order execution processor system**
*Saxena, N.; Chien Chen; Swami, R.; Osone, H.; Thusoo, S.; Lyon, D.; Chang, D.; Dharmaraj, A.; Patkar, N.; Lu, Y.; Chia, B.;*
Fault-Tolerant Computing, 1995. FTCS-25. Digest of Papers., Twenty-Fifth International Symposium on , 27-30 June 1995
Pages:464 - 471

[Abstract] [PDF Full-Text (676 KB)] **IEEE CNF**

---

Home | Log-out | Journals | Conference Proceedings | Standards | Search by Author | Basic Search | Advanced Search | Join IEEE | Web Account | New this week | OPAC Linking Information | Your Feedback | Technical Support | Email Alerting | No Robots Please | Release Notes | IEEE Online Publications | Help | FAQ| Terms | Back to Top